

	Type	L #	Hits	Search Text	DBs	Time Stamp
1	BRS	L1	6066	thread same suspend\$	US- PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2005/07/22 14:39
2	BRS	L2	60	1 and monitor with (resume or resumption)	US- PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2005/07/22 14:39
3	BRS	L3	29	2 and memory adj access	US- PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2005/07/22 14:40
4	BRS	L4	24	3 and monitor with address	US- PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2005/07/22 14:40
5	BRS	L5	23	4 and interrupt	US- PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2005/07/22 14:40
6	BRS	L6	23	5 and flush\$	US- PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2005/07/22 14:42

	Type	L #	Hits	Search Text	DBs	Time Stamp
8	BRS	L8	4	7 and break adj event	US- PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2005/07/22 14:43
9	BRS	L9	4	7 and break	US- PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2005/07/22 15:52
10	BRS	L10	19	7 not 9	US- PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2005/07/22 16:06
11	IS&R	L11	4	((("6493741") or ("5530597"))).PN.	US- PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2005/07/22 16:08
12	BRS	L12	0	712/27,235,244,219,32	US- PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2005/07/22 16:14
13	IS&R	L13	1308	(712/27,235,244,219,32).CCLS.	US- PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2005/07/22 16:18

14	BRS	L14	27	13 and 1	US- PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2005/07/22 16:14
----	-----	-----	----	----------	--	---------------------

	Type	L #	Hits	Search Text	DBs	Time Stamp
15	IS&R	L15	1781	(712/227,235,244,219,32).CCLS.	US- PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2005/07/22 16:18
16	IS&R	L16	0	("015and1").PN.	US- PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2005/07/22 16:19
17	BRS	L17	37	15 and 1	US- PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2005/07/22 16:19



USPTO

[Subscribe \(Full Service\)](#) [Register \(Limited Service, Free\)](#) [Login](#)

Search: ☒ The ACM Digital Library ☐ The Guide

thread and suspend and resume and monitor and "memory ac

SEARCH



[Feedback](#) [Report a problem](#) [Satisfaction survey](#)

Terms used

thread and suspend and resume and monitor and memory access and interrupt and implicit operand and writeback and break

Found 2,986 of 158,639

Sort results by

relevance ☒



[Save results to a Binder](#)

[Try an Advanced Search](#)

Try this search in [The ACM Guide](#)

Display results

expanded form ☒



[Search Tips](#)

☐ Open results in a new window

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

Best 200 shown

Relevance scale ☐ ☐ ☐ ☐ ☐

1 [The family of concurrent logic programming languages](#)

Ehud Shapiro

September 1989 **ACM Computing Surveys (CSUR)**, Volume 21 Issue 3

Full text available: [pdf\(9.62 MB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Concurrent logic languages are high-level programming languages for parallel and distributed systems that offer a wide range of both known and novel concurrent programming techniques. Being logic programming languages, they preserve many advantages of the abstract logic programming model, including the logical reading of programs and computations, the convenience of representing data structures with logical terms and manipulating them using unification, and the amenability to metaprogrammin ...

2 [Experiences with building distributed debuggers](#)

Michael S. Meier, Kevan L. Miller, Donald P. Pazel, Josyula R. Rao, James R. Russell

January 1996 **Proceedings of the SIGMETRICS symposium on Parallel and distributed tools**

Full text available: [pdf\(1.34 MB\)](#)

Additional Information: [full citation](#), [references](#), [index terms](#)

3 [Guide for the use of the Ada Ravenscar Profile in high integrity systems](#)

Alan Burns, Brian Dobbing, Tullio Vardanega

June 2004 **ACM SIGAda Ada Letters**, Volume XXIV Issue 2

Full text available: [pdf\(548.17 KB\)](#)

Additional Information: [full citation](#), [references](#)

4 [Helper threads via virtual multithreading on an experimental itanium® 2 processor-based platform](#)

Perry H. Wang, Jamison D. Collins, Hong Wang, Dongkeun Kim, Bill Greene, Kai-Ming Chan, Aamir B. Yunus, Terry Sych, Stephen F. Moore, John P. Shen

October 2004 **Proceedings of the 11th international conference on Architectural support for programming languages and operating systems**, Volume 39 , 38 , 32 Issue 11 , 5 , 5

Full text available: [pdf\(225.47 KB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Helper threading is a technology to accelerate a program by exploiting a processor's multithreading capability to run ``assist'' threads. Previous experiments on hyper-threaded

processors have demonstrated significant speedups by using helper threads to prefetch hard-to-predict delinquent data accesses. In order to apply this technique to processors that do not have built-in hardware support for multithreading, we introduce virtual multithreading (VMT), a novel form of switch-on-event user-level ...

Keywords: DB2 database, PAL, cache miss prefetching, helper thread, itanium processor, multithreading, switch-on-event

5 Using Hardware Counters to Automatically Improve Memory Performance

Mustafa M. Tikir, Jeffrey K. Hollingsworth

November 2004 **Proceedings of the 2004 ACM/IEEE conference on Supercomputing**

Full text available:  [pdf\(152.84 KB\)](#) Additional Information: [full citation](#), [abstract](#)

In this paper, we introduce a profile-driven online page migration scheme and investigate its impact on the performance of multithreaded applications. We use lightweight, inexpensive plug-in hardware counters to profile the memory access behavior of an application, and then migrate pages to memory local to the most frequently accessing processor. Using the Dyninst runtime instrumentation combined with hardware counters, we were able to add page migration capabilities to the system without having ...

6 Process migration

September 2000 **ACM Computing Surveys (CSUR)**, Volume 32 Issue 3

Full text available:  [pdf\(1.24 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

Process migration is the act of transferring a process between two machines. It enables dynamic load distribution, fault resilience, eased system administration, and data access locality. Despite these goals and ongoing research efforts, migration has not achieved widespread use. With the increasing deployment of distributed systems in general, and distributed operating systems in particular, process migration is again receiving more attention in both research and product development. As hi ...

Keywords: distributed operating systems, distributed systems, load distribution, process migration

7 Enhancing software reliability with speculative threads

Jeffrey Oplinger, Monica S. Lam

October 2002 **Proceedings of the 10th international conference on Architectural support for programming languages and operating systems**, Volume 37 , 36 , 30 Issue 10 , 5 , 5


Full text available:  [pdf\(1.47 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#)

This paper advocates the use of a monitor-and-recover programming paradigm to enhance the reliability of software, and proposes an architectural design that allows software and hardware to cooperate in making this paradigm more efficient and easier to program. We propose that programmers write monitoring functions assuming simple sequential execution semantics. Our architecture speeds up the computation by executing the monitoring functions speculatively in parallel with the main computation. For ...

8 Virtual machine monitors: Xen and the art of virtualization

Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield

October 2003 **Proceedings of the nineteenth ACM symposium on Operating systems principles**

Full text available:  [pdf\(168.76 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Numerous systems have been designed which use virtualization to subdivide the ample resources of a modern computer. Some require specialized hardware, or cannot support

commodity operating systems. Some target 100% binary compatibility at the expense of performance. Others sacrifice security or functionality for speed. Few offer resource isolation or performance guarantees; most provide only best-effort provisioning, risking denial of service. This paper presents Xen, an x86 virtual machine monitor ...

Keywords: hypervisors, paravirtualization, virtual machine monitors

9 An on-the-fly mark and sweep garbage collector based on sliding views

Hezi Azatchi, Yossi Levanoni, Harel Paz, Erez Petrank

October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications**, Volume 38 Issue 11

Full text available:  pdf(244.12 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)


With concurrent and garbage collected languages like Java and C# becoming popular, the need for a suitable non-intrusive, efficient, and concurrent multiprocessor garbage collector has become acute. We propose a novel mark and sweep on-the-fly algorithm based on the sliding views mechanism of Levanoni and Petrank. We have implemented our collector on the Jikes Java Virtual Machine running on a Netfinity multiprocessor and compared it to the concurrent algorithm and to the stop-the-world collector ...

Keywords: concurrent garbage collection, garbage collection, memory management, on-the-fly garbage collection, runtime systems

10 Tempest and typhoon: user-level shared memory

S. K. Reinhardt, J. R. Larus, D. A. Wood

April 1994 **ACM SIGARCH Computer Architecture News , Proceedings of the 21ST annual international symposium on Computer architecture**, Volume 22 Issue 2

Full text available:  pdf(1.44 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Future parallel computers must efficiently execute not only hand-coded applications but also programs written in high-level, parallel programming languages. Today's machines limit these programs to a single communication paradigm, either message-passing or shared-memory, which results in uneven performance. This paper addresses this problem by defining an interface, *Tempest*, that exposes low-level communication and memory-system mechanisms so programmers and compilers can customize policies ...

11 EROS: a fast capability system

Jonathan S. Shapiro, Jonathan M. Smith, David J. Farber

December 1999 **ACM SIGOPS Operating Systems Review , Proceedings of the seventeenth ACM symposium on Operating systems principles**, Volume 33 Issue 5

Full text available:  pdf(1.83 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

EROS is a capability-based operating system for commodity processors which uses a single level storage model. The single level store's persistence is transparent to applications. The performance consequences of support for transparent persistence and capability-based architectures are generally believed to be negative. Surprisingly, the basic operations of EROS (such as IPC) are generally comparable in cost to similar operations in conventional systems. This is demonstrated with a set of microbenchmarks ...

12 Experience Using Multiprocessor Systems—A Status Report

Anita K. Jones, Peter Schwarz

June 1980 **ACM Computing Surveys (CSUR)**, Volume 12 Issue 2

Full text available:  pdf(4.48 MB) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

13 Maintaining Consistency and Bounding Capacity of Software Code Caches

Derek Bruening, Saman Amarasinghe

March 2005 **Proceedings of the international symposium on Code generation and optimization CGO '05**

Full text available: [pdf\(253.56 KB\)](#) Additional Information: [full citation](#), [abstract](#)

Software code caches are becoming ubiquitous, in dynamic optimizers, runtime tool platforms, dynamic translators, fast simulators and emulators, and dynamic compilers. Caching frequently executed fragments of code provides significant performance boosts, reducing the overhead of translation and emulation and meeting or exceeding native performance in dynamic optimizers. One disadvantage of caching, memory expansion, can sometimes be ignored when executing a single application. However, as optimiz ...

14 A survey of commercial parallel processors

Edward Gehringer, Janne Abullarade, Michael H. Gulyan

September 1988 **ACM SIGARCH Computer Architecture News**, Volume 16 Issue 4

Full text available: [pdf\(2.96 MB\)](#) Additional Information: [full citation](#), [abstract](#), [citations](#), [index terms](#)

This paper compares eight commercial parallel processors along several dimensions. The processors include four shared-bus multiprocessors (the Encore Multimax, the Sequent Balance system, the Alliant FX series, and the ELXSI System 6400) and four network multiprocessors (the BBN Butterfly, the NCUBE, the Intel iPSC/2, and the FPS T Series). The paper contrasts the computers from the standpoint of interconnection structures, memory configurations, and interprocessor communication. Also, the share ...

15 Hive: fault containment for shared-memory multiprocessors

J. Chapin, M. Rosenblum, S. Devine, T. Lahiri, D. Teodosiu, A. Gupta

December 1995 **ACM SIGOPS Operating Systems Review , Proceedings of the fifteenth ACM symposium on Operating systems principles**, Volume 29 Issue 5

Full text available: [pdf\(1.90 MB\)](#) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

16 Multiprocessor enhancements of the SimpleScalar tool set

Naraig Manjikian

March 2001 **ACM SIGARCH Computer Architecture News**, Volume 29 Issue 1

Full text available: [pdf\(829.23 KB\)](#) Additional Information: [full citation](#), [abstract](#), [citations](#), [index terms](#)

This paper describes multiprocessor enhancements of the SimpleScalar tool set. The core simulation code has been modified to support multiprocessing, and a run-time library has been introduced for thread creation and synchronization. Measurements using the SPLASH-2 parallel benchmark suite [13] indicate that the multiprocessor enhancements introduce simulation overhead of 30%-50% relative to the original uniprocessor simulator. An idealized multiprocessor cache simulator has also been developed, ...

17 Kill-safe synchronization abstractions

Matthew Flatt, Robert Bruce Findler

June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation**, Volume 39 Issue 6

Full text available: [pdf\(138.55 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

When an individual task can be forcefully terminated at any time, cooperating tasks must communicate carefully. For example, if two tasks share an object, and if one task is terminated while it manipulates the object, the object may remain in an inconsistent or frozen state that incapacitates the other task. To support communication among terminable tasks, language run-time systems (and operating systems) provide kill-safe abstractions for inter-task communication. No kill-safe guarantee is avai ...

18 A survey of rollback-recovery protocols in message-passing systems

E. N. (Mootaz) Elnozahy, Lorenzo Alvisi, Yi-Min Wang, David B. Johnson

September 2002 **ACM Computing Surveys (CSUR)**, Volume 34 Issue 3

Full text available:  pdf(549.68 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

This survey covers rollback-recovery techniques that do not require special language constructs. In the first part of the survey we classify rollback-recovery protocols into *checkpoint-based* and *log-based*. *Checkpoint-based* protocols rely solely on checkpointing for system state restoration. Checkpointing can be coordinated, uncoordinated, or communication-induced. *Log-based* protocols combine checkpointing with logging of nondeterministic events, encoded in tuples call ...

Keywords: message logging, rollback-recovery

19 An object-based programming model for shared data

Gail E. Kaiser, Brent Hailpern

April 1992 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 14 Issue 2

Full text available:  pdf(3.28 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#), [review](#)

The classical object model supports private data within objects and clean interfaces between objects, and by definition does not permit sharing of data among arbitrary objects. This is a problem for real-world applications, such as advanced financial services and integrated network management, where the same data logically belong to multiple objects and may be distributed over multiple nodes on the network. Rather than give up the advantages of encapsulated objects in modeling real-world en ...

Keywords: coordination language, daemons, financial applications, object-based, real-time, sharing

20 Multithreading in C++

John English

April 1995 **ACM SIGPLAN Notices**, Volume 30 Issue 4

Full text available:  pdf(679.96 KB) Additional Information: [full citation](#), [abstract](#), [citations](#), [index terms](#)

This paper describes the design and implementation of a C++ class library which provides timesliced multithreading capabilities for C++ programs. The implementation described here runs under MS-DOS on an IBM PC, but the library could easily be reimplemented for other systems. The library and its source code are freely available by anonymous FTP from a number of sites.

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2005 ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)